САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА ТЕОРИИ СИСТЕМ УПРАВЛЕНИЯ ЭЛЕКТРОФИЗИЧЕСКОЙ АППАРАТУРОЙ

Мащинский Николай Сергеевич

Выпускная квалификационная работа бакалавра

Моделирование радиоэлектронного устройства и проверяющих тестовых воздействий

Направление 010900

Прикладные математика и физика

Научный руководитель,
доктор физ.-мат. наук,
профессор
Овсянников Д. А.

Рецензент,
кандидат тех. наук,
доцент
Гришкин В. М.

Санкт-Петербург
2016

Оглавление

Введение

Сегодня человечество сильно зависит от электронных устройств, которые главным образом используются в таких областях цифровой техники, как автоматизация, вычислительная техника, робототехника, микропроцессорные измерительные приборы, спутниковая связь и телевидение. Все эти устройства строятся на единой базе элементов, в которой содержатся как микросхемы, выполняющие простые логические операции, так и сложные программируемые кристаллы, состоящие из тысяч и миллионов логических элементов.

Для качественного функционирования радиоэлектронных устройств необходимо их тестирование на всех этапах серийного производства. Раньше это делалось вручную — человек щупами проверял работу всех микросхем и сигнальных линий. По статистике, из-за человеческого фактора пропускалась четверть всех дефектов. Сейчас широко распространено автоматизированное тестовое оборудование, позволяющее чрезвычайно ускорить нахождение бракованных изделий и обнаружение в них неисправностей, что значительно уменьшает возможность выпуска неработоспособных электронных устройств и снижает стоимость исправления дефектов.

Современное состояние научных исследований

В процессе производства радиоэлектронных устройств применяются следующие типы автоматизированного тестового оборудования:

- визуальный автоматизированный контроль,
- внутрисхемное тестирование,
- периферийное/граничное сканирование,
- функциональное тестирование.

Системы автоматизированного визуального контроля используются для предварительной проверки монтажа и качества печатных плат. Они с высокой скоростью обнаруживают на поверхности печатных плат такие дефекты, как истончения припоя или короткие замыкания, и определяют неправильно установленные микросхемы или резисторы. Подобные системы также могут использовать в процессе диагностики рентгеновское излучение, что позволяет им исследовать недоступные для обычных оптических технологий места электронного изделия, однако для этого требуется сложное и дорогостоящее оборудование.

При внутрисхемном тестировании нет необходимости подводить питание ко всему тестируемому устройству, так как сигналы на разъемы внутренних элементов подаются и считываются при помощи контактных иголок тестера. Таким способом можно выполнить предварительную отбраковку печатных плат с неправильным монтажом резисторов или опасными короткими замыканиями, которые после включения питания могли бы привести к значительным повреждениям устройства. Учитывающая индивидуальные особенности изделия матрица контактов (устаревшее название «ложе гвоздей») [1–2] обеспечивает высокую производительность процесса диагностики, однако требует больших затрат при переходе от одного типа плат к другому. Системы «летающих щупов» [3–4] более

универсальны, но обладают невысокой производительностью. Технология внутрисхемного тестирования не применима для многослойных печатных плат, электронных устройств с относительно маленькими микросхемами и при наличии на поверхности объекта контроля защитного лака.

Для проведения граничного сканирования при проектировании электронных изделий необходима их предварительная подготовка — использование в микросхемах стандарта JTAG, правильное соединение и вывод JTAG-портов на внешние разъемы. Принцип граничного сканирования заключается в размещении по границам устройства последовательного сдвигового регистра, ячейки которого находятся между краевыми разъемами электронного устройства и логическим ядром. После подачи на тестовый вход определенной двоичной последовательности, полученный на тестовом выходе результат сравнивается с эталонным. Если в некоторых элементах устройства отсутствует структура JTAG, то их работоспособность может определяться косвенным образом [5–13]. Недостатками этого метода являются его непригодность для проверки аналоговых компонентов и невозможность проверки качества связей, что может существенно повлиять на работу высокоскоростных электронных устройств.

Функциональное тестирование, как следует из названия, выполняет проверку функциональности электронных изделий на соответствие заложенной в них спецификации. Используя программную модель объекта контроля, можно предсказать поведение устройства с возможными неисправностями и проверить это предположение при помощи измерений на реальном устройстве [14–28]. Несмотря на то, что функциональное тестирование иногда не позволяет точно определить неисправные компоненты и требует разработку программного обеспечения, в большинстве случаев применение технологии оправдывается максимальным покрытием проверяемых компонентов устройства и коротким временем тестирования.

Проанализировав современные методы диагностирования неисправностей радиоэлектронных устройств, можно сделать вывод о том, что по сравнению с другими технологиями функциональное тестирование обладает следующими преимуществами:

- универсальность применения (от компонентов устройства не требуется поддержка стандарта JTAG);
- нет зависимости от способа производства радиоэлектронного устройства (внутрисхемное тестирование может эффективно применяться только к однослойным печатным платам);
- относительная дешевизна (по сравнению с системами автоматизированного визуального контроля и внутрисхемного тестирования);
- обладает высокой скоростью тестирования, что принципиально при серийном производстве электронных изделий.

Используя разработанную в СПбГУ систему автоматизированного проектирования тестов «SimTest», можно автоматизировать процесс составления тестовой программы для проверки работоспособности и выявления неисправностей радиоэлектронного устройства [29–32]. Тестовая программа описывает соответствие входных и выходных сигналов на краевых разъемах исправного изделия и может быть загружена в установку тестового контроля, являющуюся «связующим звеном» между поведенческой программной моделью и тестируемым устройством [33–34].

Постановка задачи

Целью работы является написание и отладка программной модели цифрового радиоэлектронного устройства Субблок 1Э3 и составление с помощью САПР «SimTest» тестовой программы, осуществляющей контроль работоспособности и диагностирование неисправностей внутренних связей и элементов объекта контроля. Тестируемое устройство состоит из 37 логических элементов, содержит 72 внутренние связи и несколько аналоговых компонентов, работу которых также нужно учесть.

Задачи, необходимые для достижения поставленной цели:

1. разработка и тестирование программных моделей внутренних элементов объекта контроля;
2. получение программной модели всего устройства;
3. составление последовательности тестовых воздействий.

Составленная тестовая программа должна обладать наибольшим возможным покрытием объекта контроля (должна активировать максимальное число компонентов и внутренних сигнальных линий), проверять работоспособность внутренних элементов, проверять целостность связей между компонентами и отсутствие дополнительных «паразитных» связей внутри устройства, а также изменять состояние выходных разъемов тестируемого устройства.

Технология разработки тестовых программ для радиоэлектронных цифровых устройств

При применении функционального тестирования, объект контроля представляется «черным ящиком», имеющим краевые выводы и обладающего некоторой функциональностью. На входные разъемы подаются сигналы, состоящие из логических нулей и единиц, а с выходных контактов считывается реакция устройства на входные воздействия. При этом предполагается, что объект контроля разработан в точном соответствии со своей спецификацией и в случае его корректной реализации обладает надежной работоспособностью.

Так как алгоритм работы электронного устройства определяется функциональностью содержащихся в нем компонентов и связывающими линиями сигналов, то тестовая программа может быть составлена без информации о функциональных особенностях устройства и при отсутствии самого объекта контроля.

На рисунке 1 отображены основные этапы процесса создания тестовой программы.

Этап 1. Программная модель объекта контроля является совокупностью программных моделей компонентов устройства и главного модуля. Разработка программных моделей компонентов основывается на составлении алгоритмов работы внутренних элементов устройства на HDL языке описания аппаратных средств. Главный модуль представляет собой описание внутренней структуры устройства, содержащее программные модели всех компонентов и связей между ними, которые являются аналогами сигнальных линий физического устройства.

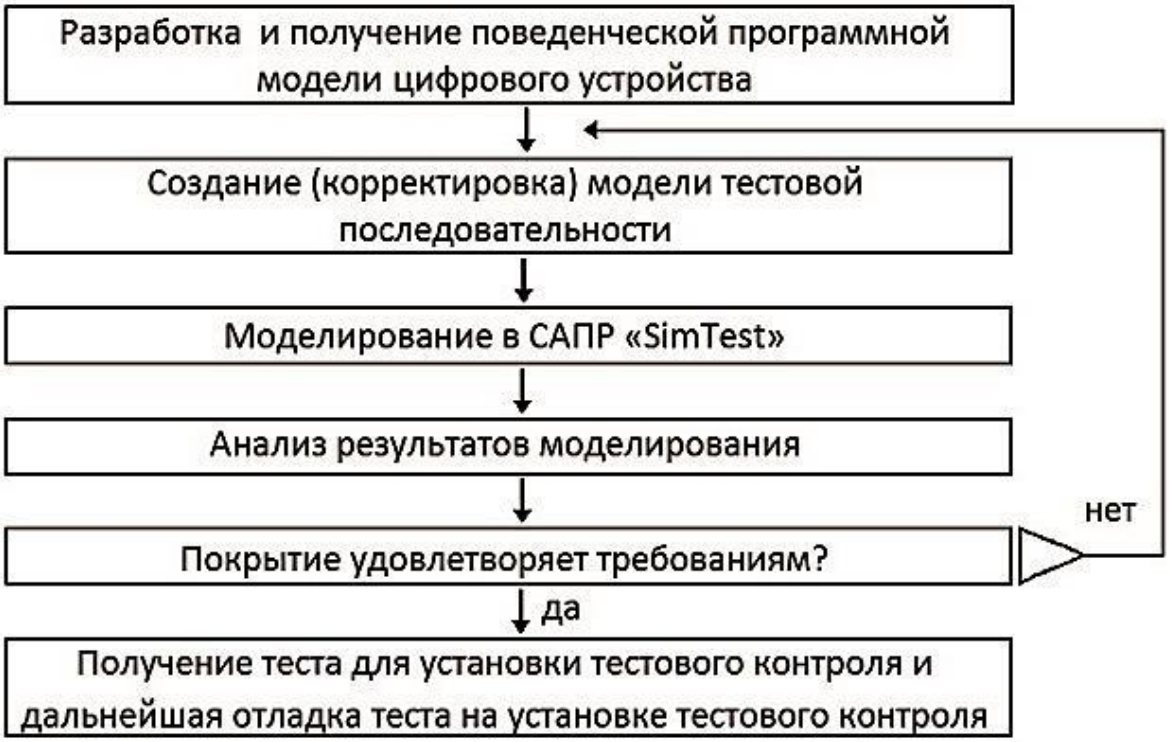

Рисунок 1. Основные этапы создания тестовой программы.

Этап 2. После составления программной модели происходит поиск временной последовательности значений сигналов, которая будет тестировать внутренние элементы и активировать сигнальные линии устройства. Зная алгоритм работы элементов и структуру сигнальных линий, оператор подбирает входные воздействия таким образом, чтобы получить на входах нужного элемента набор сигналов, проверяющий его исправность. Затем входные воздействия подбираются так, чтобы последовательность сигналов, полученная на выходах проверяемого элемента, достигла краевых выходов. Анализируя значения сигналов на выходах тестируемого устройства, можно будет сделать вывод о работоспособности устройства в целом и локализировать область возможной неисправности.

Этап 3. САПР «SimTest» «подает» тестовую последовательность на входы программной модели и записывает в файл результаты моделирования — временные состояния всех сигнальных линий. На основе анализа файла результатов система вычисляет покрытие — процентное отношение изменивших свое состояние в ходе выполнения теста сигнальных

линий к общему числу таких линий в устройстве. При создании последовательности тестовых воздействий необходима активация как можно большего числа сигнальных линий, и, при возможности, достижение стопроцентного покрытия. Также наличие файла результатов позволяет изменять последовательность входных воздействий необходимым для тестирования образом без использования физического устройства, проверять правильность реализации функционала элементов объекта контроля и корректность связей между ними в программной модели.

Этап 4. После достижения требуемого уровня покрытия тестовая программа отлаживается на заведомо исправном устройстве с помощью установки тестового контроля. Это необходимо для проверки соответствия между созданной программной моделью и реальным устройством. Если при подаче тестовой последовательности возникают различия между значениями выходных сигналов объекта контроля и значениями, полученными при помощи использования программной модели, то происходит корректировка программной модели и повторение предыдущих этапов создания тестовой программы.

Корректировка программной модели значительно упрощается в том случае, когда технология производства электронного устройства позволяет осуществить прямой доступ к контактным площадкам. Используя синхронизированный с установкой тестового контроля осциллограф, можно получить информацию о состоянии сигнальных линий устройства. Сравнивая эту информацию со значениями соответствующих сигнальных линий в файле результатов моделирования, оператор быстрее определяет некорректность в структуре программной модели или в описании алгоритма работы компонента.

После этого тестовая программа готова к контролю и диагностике серийно производимых радиоэлектронных цифровых устройств.

Для устройств, содержащих элементы с памятью или программной логикой, необходимо разрабатывать дополнительные тесты, которые проверяли бы работоспособность этих сложных элементов. Такие тесты называются диагностическими.

Существует иной подход к тестированию цифровых устройств — на заведомо исправное устройство подать всевозможные последовательности входных воздействий, зафиксировать последовательности выходных сигналов устройства и полученным «портретом» тестировать другое устройство на наличие несовпадений в последовательностях выходных сигналов, что означало бы неисправность тестируемого устройства. Но при таком подходе есть вероятность испортить заведомо исправное устройство при составлении тестового «портрета» и нет никаких гарантий, что подаваемые входные воздействия активируют все внутренние линии сигналов и проверяют элементы устройства необходимым для этого образом.

Программное моделирование элементов устройства

Объект контроля Субблок 1Э3 состоит из микросхем 133LA2, 133LA3, 133LA4, 133LA7, 133LA8 (выполняют булевы функции 8И–НЕ, 2И–НЕ, 3И–НЕ, 4И–НЕ и 2И–НЕ соответственно), 133ТМ2 (содержит 2 идентичных синхронных D–триггера с дополняющими выходами), и 133IE5 (четырехразрядный двоичный счетчик). Ниже представлены условно–графические отображения (УГО), таблицы работы и разработанные программные модели для этих элементов на языке описания аппаратных средств Verilog HDL.

1) Микросхема 133LA4 содержит три идентичных логических элемента, выполняющих булеву функцию 3И–НЕ (рисунок 2): $out = \overline{in1 \,\&\, in2 \,\&\, in3}$.

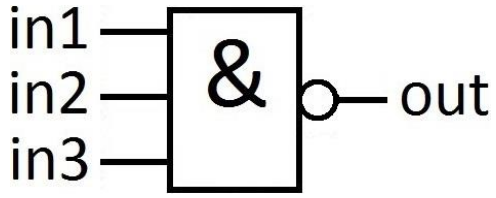

Рисунок 2. УГО логического элемента 3И–НЕ.

in1	0	1	0	1	0	1	0	1
in2	0	0	1	1	0	0	1	1
in3	0	0	0	0	1	1	1	1
out	1	1	1	1	1	1	1	0

Таблица 1. Таблица истинности для элемента 3И–НЕ.

Так как логический элемент 3И–НЕ является комбинационным (не содержит ячеек памяти, состояние выходов однозначно определяется набором входных сигналов), то логика его работы может быть описана при помощи таблицы 1.

Программная модель этого элемента на языке Verilog HDL описывается следующим образом:

```
module icp_133la4
(
        input   in1, in2, in3,
        output out
) ;
assign out = ~(in1 & in2 & in3);
endmodule
```

Ключевое слово «assign» выполняет непрерывное, на каждом такте модельного времени, присваивание значения функции, находящейся в правой части от знака «=», сигнальному проводу *out*, который одновременно является выходом элемента 133LA4.

Для микросхем 133LA2, 133LA3, 133LA7 и 133LA8, выполняющих булевы функции 8И–НЕ, 2И–НЕ, 4И–НЕ и 2И–НЕ соответственно, условно-графические отображения, таблицы истинности, описывающие логику их работы, и программные модели на языке Verilog выглядят аналогично. Отличие состоит только в количестве входных сигналов.

2) Микросхема 133ТМ2 содержит два идентичных независимых D–триггера (рисунок 3), срабатывающих по положительному фронту тактового сигнала *C*, то есть при переходе сигнала *C* из состояния логического нуля в состояние логической единицы.

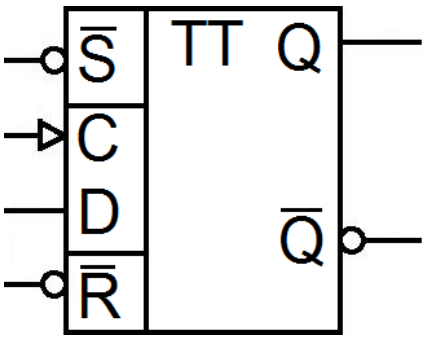

Рисунок 3. УГО D–триггера.

Значение 0 на входах установки $\overline{S}$ или сброса $\overline{R}$ устанавливает выходы триггера в соответствующее состояние вне зависимости от состояния на других входах (*C* и *D*). При наличии на входах установки $\overline{S}$ и сброса $\overline{R}$

значения 1, для правильной работы триггера требуется предварительная установка информации по входу данных D относительно положительного фронта тактового сигнала C, а также соответствующая выдержка информации после подачи положительного фронта синхросигнала C [35].

Входы				Выходы	
$\bar{S}$	$\bar{R}$	C	D	Q	$\overline{Q}$
0	1	x	x	1	0
1	0	x	x	0	1
0	0	x	x	err	err
1	1	Г	1	1	0
1	1	Г	0	0	1
1	1	0	x	Q_0	$\overline{Q_0}$

Таблица 2. Таблица работы D–триггера.

В таблице 2 представлена логика работы D–триггера, где x — любое состояние сигнала, 0 или 1; Г — положительный фронт сигнала, переход сигнала из состояния 0 в состояние 1; Q_0, $\overline{Q_0}$ — предыдущее состояние выходов Q, $\overline{Q}$; err — неопределенное состояние выхода.

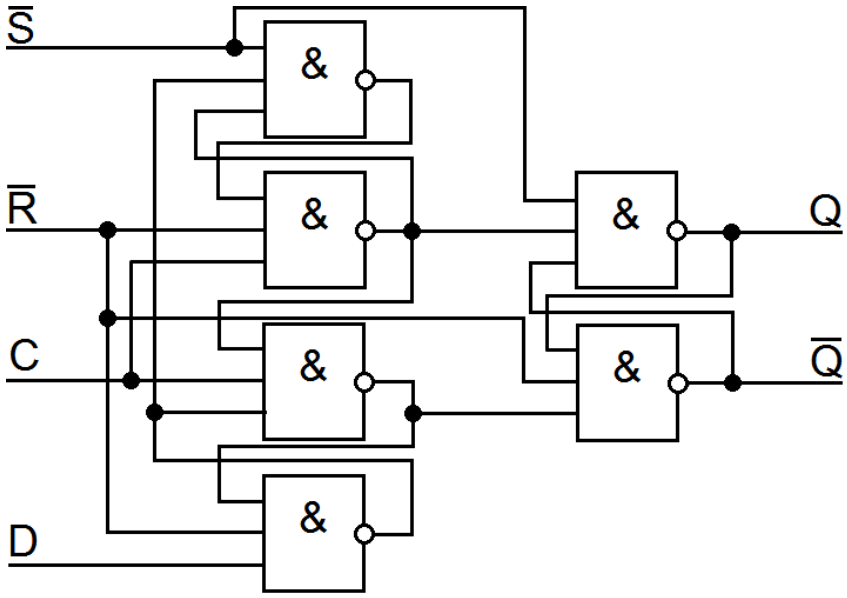

Рисунок 4. Функциональная схема D–триггера.

Представленная на рисунке 4 функциональная схема D–триггера состоит из шести логических элементов 3И–НЕ, рассмотренных ранее. Через *P1, ..., P6* обозначены сигнальные линии, присоединенные к выходам элементов *D1, ..., D6* соответственно.

С учетом функциональной схемы, программную модель D–триггера можно описать следующим образом:

```
module icp_133tm2
(
        input   D, C, R_n, S_n,
        output Q, Q_n
);
wire   P1;
wire   P2;
wire   P3;
wire   P4;
wire   P5;
wire   P6;

icp_133la4  b2v_D1(
        .in1(S_n),
        .in2(P4),
        .in3(P2),
        .out(P1));

icp_133la4  b2v_D2(
        .in1(P1),
        .in2(R_n),
        .in3(C),
        .out(P2));

icp_133la4  b2v_D3(
        .in1(P2),
        .in2(C),
        .in3(P4),
        .out(P3));
```

```
icp_133la4  b2v_D4(
        .in1(P3),
        .in2(R_n),
        .in3(D),
        .out(P4));

icp_133la4  b2v_D5(
        .in1(S_n),
        .in2(P2),
        .in3(P6),
        .out(P5));

icp_133la4  b2v_D6(
        .in1(P5),
        .in2(R_n),
        .in3(P3),
        .out(P6));

assignQ = P5;
assignQ_n = P6;
endmodule
```

Зная описание функциональности элемента или его таблицу работы, можно реализовать программную модель компонента, не учитывая его внутреннюю структуру, представленную в функциональной схеме. Хотя такое описание требует более глубоких познаний языка Verilog HDL, оно почти втрое короче предыдущего.

```
module icp_133tm2
(
        input s0_n, c0, d0, r0_n,
        output q0,
        output q0_n
);
reg temp = 0;
assign q0_n = ~q0;
```

```
always@(posedge c0 or negedge r0_n or negedge s0_n)
begin
        if(s0_n == 1'b0)
                temp = 1'b1;
        else
                if(r0_n ==1'b0)
                        temp = 1'b0;
                else
                        temp = d0;
end

assign q0 = (s0_n == 1'b1)? temp : 1'b1;
endmodule
```

В отличие от непрерывного присваивания «assign», ключевое слово «always» позволяет выполнение операторов в блоке *begin / end* только при наступлении одного из событий, которые указываются в скобках после «@». Событие «posedge c0» означает ожидание на проводе *c0* положительного фронта сигнала. Ключевое слово «negedge» используется при ожидании на соответствующем проводе отрицательного фронта сигнала.

Поскольку в блоке операторов после ключевого «always» возможна запись значений только в регистры, то для присвоения необходимых значений проводу *q0* используется регистр *temp*.

3) Микросхема 133IE5 представляет собой четырехразрядный двоичный счетчик (рисунок 5) и содержит четыре триггера, срабатывающих по отрицательному фронту на информационных входах *A*, *B*, а также дополнительные связи, реализующие в микросхеме две секции: счетчик–делитель на 2 и трехразрядный счетчик–делитель на 8. Каждая секция может использоваться отдельно, а для получения 4-х разрядного счетчика используется внешняя связь выхода *Q1* счетчика–делителя на 2 с информационным входом *B* счетчика–делителя на восемь [35].

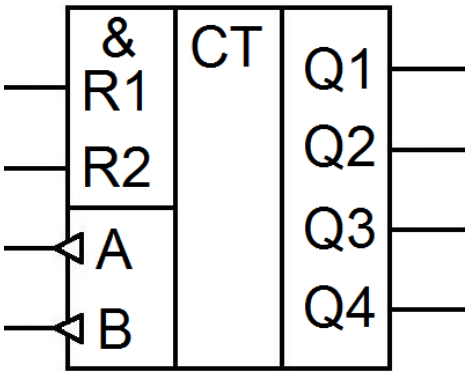

Рисунок 5. УГО четырехразрядного двоичного счетчика.

Являясь элементом с ячейками памяти, в состоянии «Счет» (таблица 3) микросхема хранит числа полученных отрицательных фронтов на информационных входах *A* и *B* и выводит эти числа в двоичном виде на выходы *Q1* и *Q2*, *Q3*, *Q4* соответственно. Также присутствуют входы сброса *R1* и *R2*, устанавливающие выходы счетчика в «ноль» (0000).

Входы		Выходы			
R1	R2	Q1	Q2	Q3	Q4
1	1	0	0	0	0
0	x	Счет			
x	0	Счет			

Таблица 3. Таблица работы микросхемы 133IE5.

Функциональность этого элемента на языке Verilog HDL можно описать следующим образом:

```
module ic_133ie5
(
        input wire a, b, r1, r2,
        output wire q1, q2, q3, q4
);
wire reset = r1 && r2;

reg mem1 = 0;
reg[2:0] mem2 = 0;
```

```
assign q1 = (reset == 1'b1)? 1'b0 : mem1;
always@(negedge a or posedge reset)
begin
        if(reset == 1'b1)
                mem1 = 1'b0;
        else
                mem1 = mem1 + 1'b1;
end

assign {q4, q3, q2} = (reset == 1'b1)? 3'b0 : mem2;
always@(negedge b or posedge reset)
begin
        if(reset == 1'b1)
                mem2 = 3'b0;
        else
                mem2 = mem2 + 1'b1;
end
endmodule
```

Выражение «{q4, q3, q2}» является конкатенацией — «сложением» в пределах программного модуля *ic_133ie5* трех одноразрядных проводов в одну трехразрядную сигнальную шину. Этой шине поразрядно присваивается значение трехразрядного регистра *mem2* или двоичного числа 3'b0 (или, что то же самое, 000), в зависимости от сигнала на проводе *reset*. Провод *reset* является внутренним проводом для программного модуля, и принимает значение 1 только в том случае, когда значения обоих сигналов, поступающих на входные провода *r1* и *r2*, принимают значение 1.

4) На рисунке 6 представлена часть функциональной схемы объекта контроля, содержащая аналоговые элементы. Конденсатор *C2* и резистор *R13* образуют интегрирующий элемент, обеспечивающий задержку распространения сигнала на величину $delay = 0.7RC$, где R – сопротивление резистора в Ом, C – емкость конденсатора в Ф. Это означает, что изменение сигнала от выхода элемента *D19_1* «дойдет» до входов элемента *D19_2* с опозданием на *delay* секунд.

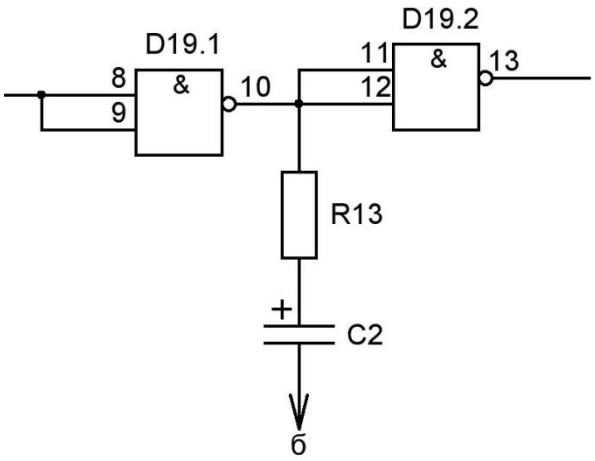

Рисунок 6. Часть функциональной схемы объекта контроля.

Программная модель задерживающего распространение сигнала элемента на языке Verilog HDL описывается следующим образом:

```
`timescale 1 us / 10 ns
module ic_delay_r13_c2_10ns
(
        input in,
        output out
);
//0.7* 160 Om *10*10^(–6) F = 0.001120 sec = 1120 us
assign #1120 out = in;
endmodule
```

Выражение «assign #1120 out = in» означает, что после изменения значения сигнала *in*, сигнал *out* будет оставаться в неизменном состоянии в течение 1120 тактов модельного времени, и только после этого ему присвоится новое значение сигнала *in*. Интервал времени между тактами моделирования задается первым параметром директивы «`timescale» и равен 1 микросекунде.

Второй параметр директивы указывает, что будет происходить округление всех задержек с точностью до 10 наносекунд. Это также означает, что если в течение менее чем 10 наносекунд придут, например, положительный и отрицательный фронты сигнала (0→1→0), то смена

значения сигнала (1) не будет отображаться в файле результатов моделирования. Несмотря на это, сама смена значения (1) может изменить состояние других элементов, к которым подключен этот сигнал. Например, неотображенный положительный фронт на входе сброса неожиданно для оператора установит выходы счетчика в «ноль».

5) На рисунке 7 представлен фрагмент функциональной схемы объекта контроля — транзисторная матрица 1НТ251, состоящая из четырех биполярных транзисторов *DA1.1, ..., DA1.4*. Вход *а* является источником постоянного напряжения, соответствующего логической единице, *б* — источником постоянного напряжения, соответствующего логическому нулю. Выводы *IN* и *OUT* представляют собой изменяемые сигнальные линии.

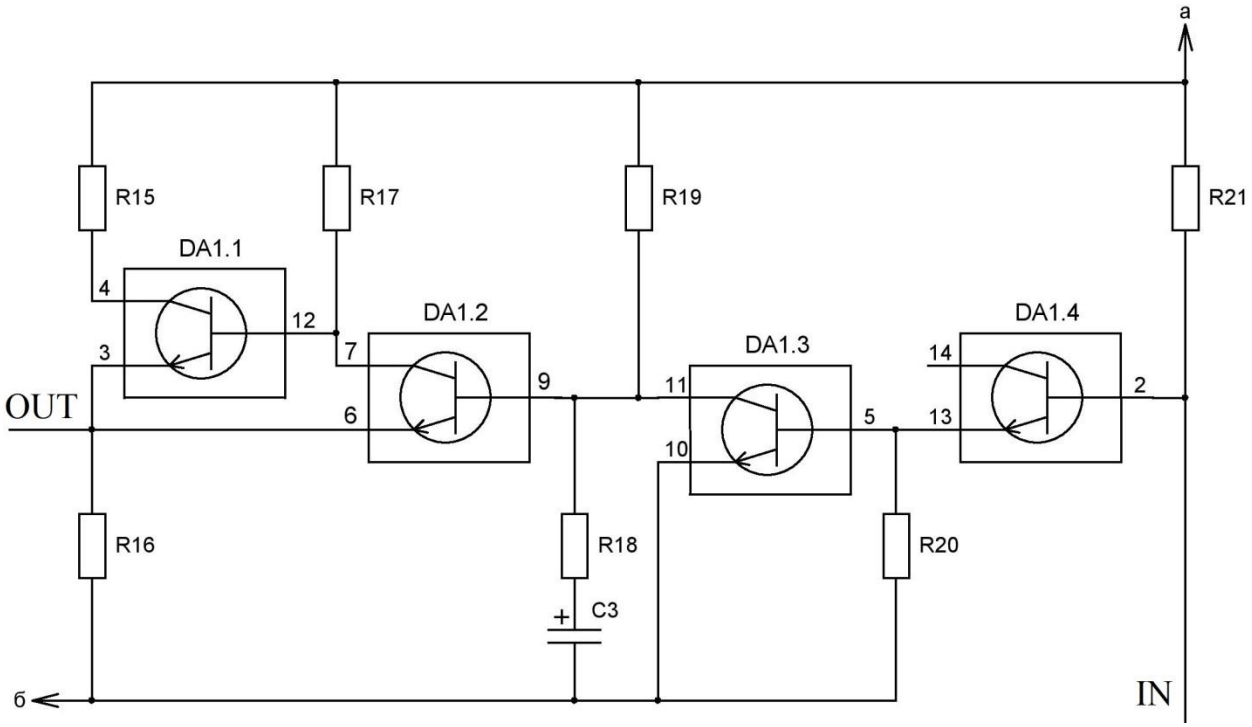

Рисунок 7. Транзисторная матрица 1НТ251.

Входной сигнал *IN* поступает на базу транзистора *DA1.4*, снимается с его эмиттера, поступает на базу транзистора *DA1.3*, снимается с его коллектора и через интегрирующий элемент поступает на базу транзистора *DA1.2*. Таким образом, транзистор *DA1.4* повторяет поступивший на базу сигнал, а *DA1.3* — инвертирует его.

Резистор *R17* обладает сопротивлением 150 кОм, а резистор *R15* — 390 Ом, поэтому вкладом в сигнал *OUT* током от эмиттера транзистора *DA1.2* можно пренебречь. Однако напряжение на коллекторе транзистора *DA1.2* будет управлять током через транзистор *DA1.1*. — при отрицательном напряжении на коллекторе *DA1.2* ток через транзистор *DA1.1* идти не будет и на выходе *OUT* будет низкий уровень напряжения, и наоборот, при положительном напряжении на коллекторе *DA1.2* ток через транзистор *DA1.1* будет проходить и на выходе *OUT* будет высокий уровень напряжения. Таким образом, можно сделать вывод о том, что транзистор *DA1.2* представляет инвертор поступающего на его базу сигнала, а *DA1.1* — повторителем.

Программные модели повторителей и инверторов описываются тривиально, а внутренняя структура программной модели транзисторной матрицы, с учетом интегрирующего элемента, будет выглядеть следующим образом:

```
module ic_1ht251
(
        input   IN,
        output OUT
) ;
wire DA1_P13;
wire DA1_P11;
wire R18_C3_P1;
wire DA1_P7;
wire DA1_P3;

ic_repeater   b2v_DA1_4(
        .in(IN),
        .out(DA1_P13));

ic_invertor   b2v_DA1_3(
        .in(DA1_P13),
        .out(DA1_P11));
```

```
ic_delay_r18_c3_10ns   b2v_R18_C3(
        .in(DA1_P11),
        .out(R18_C3_P1));

ic_invertor   b2v_DA1_2(
        .in(R18_C3_P13),
        .out(DA1_P7));

ic_repeater   b2v_DA1_1(
        .in(DA1_P7),
        .out(DA1_P3));

assign OUT = DA1_P3;
endmodule
```

Обзор сложностей, возникающих на этапе разработки программных моделей элементов устройства

Несмотря на то, что функциональная схема конкретной микросхемы может быть довольно сложной и состоять из многих логических элементов, прямого доступа к этим элементам нет. То есть невозможно физически проверить целостность внутренних связей микросхемы или заменить какой–либо компонент, представленный на функциональной схеме.

Таким образом, моделирование всех компонент и связей между ними в полном соответствии с функциональной схемой микросхемы может быть не только избыточным, но и нежелательным, так как тратится больше времени на подробное составление такой программной модели, ее отладку, составление теста для должной проверки всех внутренних компонент этой модели, что также влечет увеличение количества тактов модельного времени и усложнение тестовой программы. Альтернативным вариантом, который и был использован при моделировании функциональности D–триггера 133TM2, счетчика–делителя 133IE5 и транзисторной матрицы 1HT251, представляется разработка программной модели, реализующей алгоритм работы микросхемы и обрабатывающий входные последовательности сигналов точно так же, как и описанная в полном соответствии с функциональной схемой модель.

Хотелось бы отметить недостаток подобного подхода, хотя этот недостаток так же в некоторой степени присутствует при полном моделировании всех компонент микросхемы. Функциональная схема объекта контроля содержит только название микросхем, входящих в состав этого устройства. Функциональную схему и таблицу режимов работы для каждого элемента приходится искать самостоятельно. Существует много изданий по схемотехнике и справочников отечественных цифровых микросхем и их зарубежных аналогов. Но зачастую они дают противоречивую или неполную

информацию об алгоритмах работы микросхем или содержат ошибки в функциональных схемах. Тогда приходится «досрочно» переходить к последнему этапу методики составления тестовой программы и, подавая определенные входные воздействия на объект контроля, косвенно изучать алгоритм работы отдельных микросхем и склоняться в пользу того или иного описания этих элементов.

На данный момент это является единственным недостатком метода реализации функциональности элементов, описанного в данном разделе. С каждым корректно проверенным устройством растет база программных моделей элементов, которые могут быть использованы при диагностике других объектов контроля. Кроме того, без файла результатов всех внутренних сигнальных линий устройства, который формируется системой «SimTest», проверка работоспособности объекта контроля была бы значительно более ресурсоемкой задачей.

Получение программной модели объекта контроля

Из программных моделей элементов, с учетом связей между ними в функциональной схеме объекта контроля, составляется графическое представление программной модели в САПР AlteraQuartus 2 (рисунок 8).

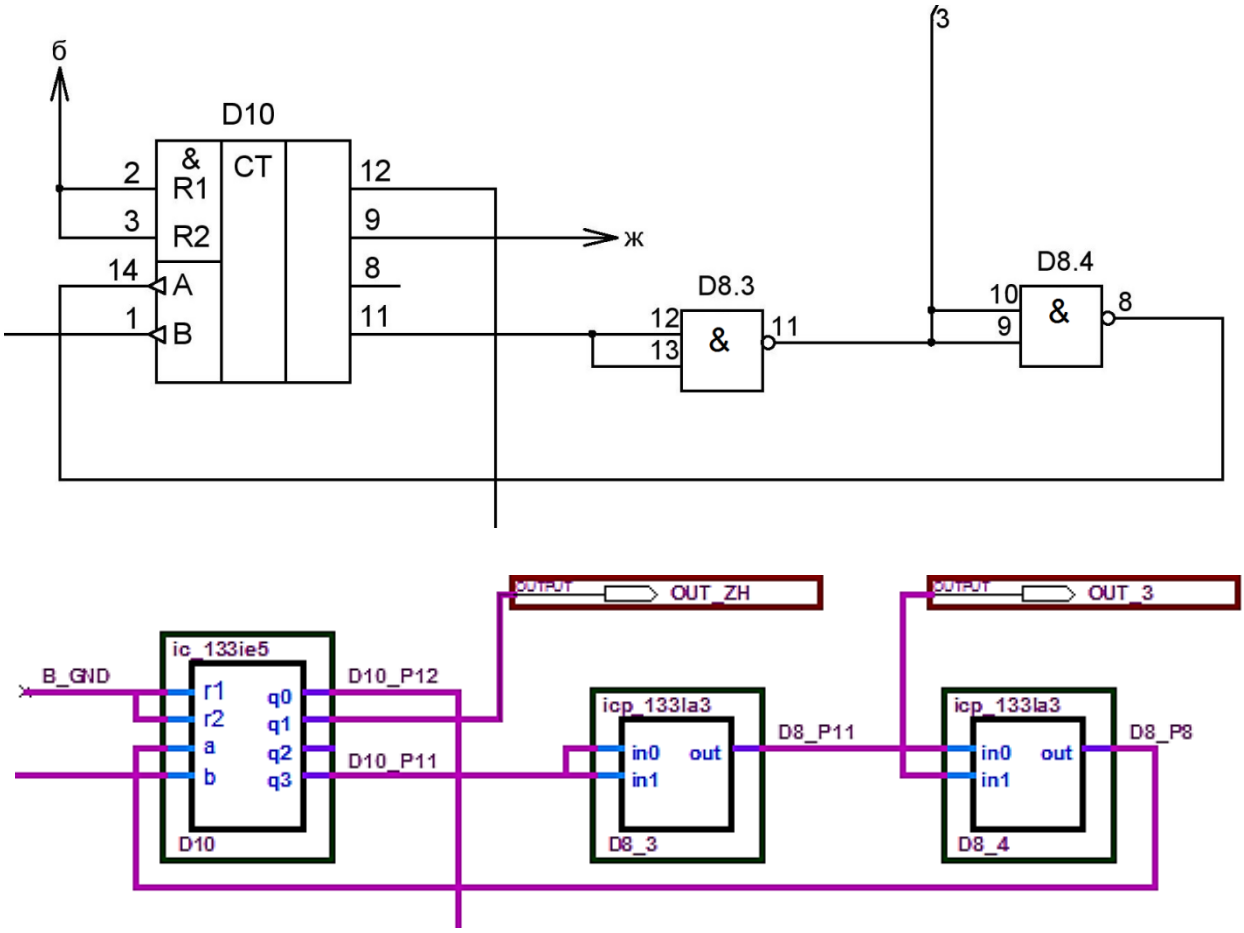

Рисунок 8. Фрагмент принципиальной схемы объекта контроля и соответствующий ему фрагмент графического представления программной модели в САПР AlteraQuartus 2.

На основе имеющихся в проекте программных моделей элементов автоматически генерируются редактируемые графические представления, отображающие их тип, количество и названия выводов. Эти представления размещаются оператором на рабочем пространстве и именуются в соответствии с принципиальной схемой объекта контроля. Затем происходит соединение выводов элементов сигнальными линиями, их именование и добавление краевых выводов объекта контроля.

Затем графическое представление всего устройства конвертируется в код, представляющий описание внутренней структуры тестируемого устройства. Ниже представлен фрагмент такого кода.

```
module Subblock_1e3_10ns
(
	IN_1,
	IN_2,
	IN_6,
	OUT_4,
	OUT_Zh,
	OUT_5,
	OUT_3,
	OUT_7,
	OUT_8,
	OUT_DIOD
);

input	IN_1;
input	IN_2;
input	IN_6;
output	OUT_4;
output	OUT_Zh;
output	OUT_5;
output	OUT_3;
output	OUT_7;
output	OUT_8;
output	OUT_DIOD;

wire	A_VCC;
wire	B_GND;
wire	D10_P11;
wire	D10_P12;
…
wire	Delay_R18_C3_P1;
wire	GEN_1_OUT;
wire	GEN_2_OUT;
```

```
ic_133ie5    b2v_D10(
        .d0(D8_P8),
        .d1(D8_P6),
        .r0(B_GND),
        .r1(B_GND),
        .q0(D10_P12),
        .q1(D10_P9),

        .q3(D10_P11));

ic_133ie5    b2v_D11(
        .d0(D10_P12),
        .d1(D11_P12),
        .r0(B_GND),
        .r1(B_GND),
        .q0(D11_P12),

        .q3(D11_P11));
…

ic_generator_10ns b2v_GEN_2(
        .out(GEN_2_OUT));

assign      OUT_4 = D8_P6;
assign      OUT_Zh = D10_P9;
assign      OUT_5 = D16_P11;
assign      OUT_3 = D8_P11;
assign      OUT_7 = D16_P3;
assign      OUT_8 = D17_P3;
assign      OUT_DIOD = D17_P6;
assign      A_VCC = 1;
assign      B_GND = 0;

endmodule
```

Также САПР AlteraQuartus 2 предоставляет возможность обнаружения ошибок в графическом представлении объекта контроля и, следовательно, в описывающем внутреннюю структуру программной модели коде.

Выявленными ошибками могут быть провода, не подключенные к какому-либо источнику сигнала, элементы, к некоторым входам или ко всем выходам которых не присоединены сигнальные линии, сигнальные линии, подключенные к нескольким источникам сигнала и невыполнимые условия в описаниях алгоритмов работы элементов.

Таким же образом создаются проекты для всех программных моделей компонентов, которые выступают в роли объекта контроля и выводы которых считаются краевыми выводами (рисунок 9); составляются файлы с описанием внутренней структуры получившихся элементов и происходит проверка их соответствия заданным алгоритмам работы с использованием САПР «SimTest».

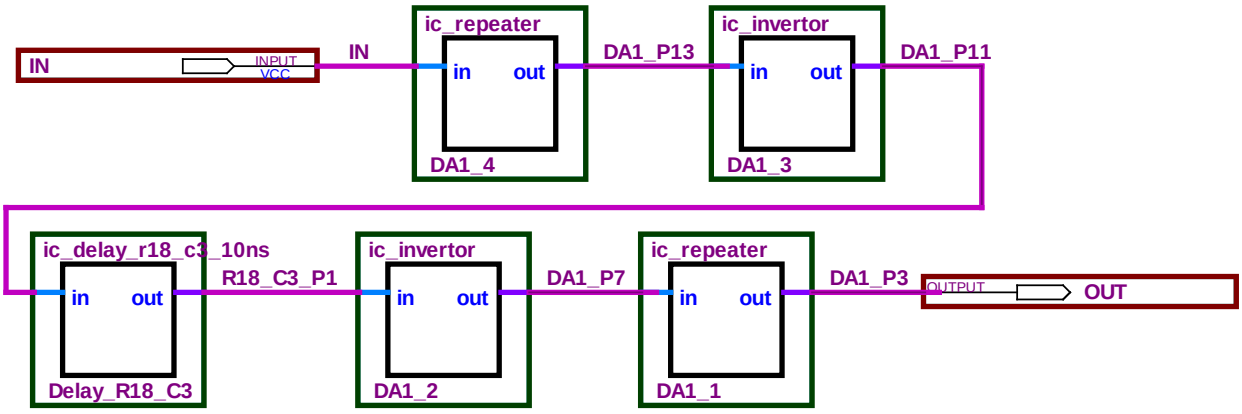

Рисунок 9. Графическое представление транзисторной матрицы 1НТ251.

Подобная проверка отличается от тестирования исходного объекта контроля только количеством выводов и длительностью временной последовательности воздействий, посредством которой определяется функциональность программной модели при всех возможных комбинациях входных сигналов.

Результаты моделирования последовательности тестовых воздействий для программной модели объекта контроля Субблок 1ЭЗ

На рисунке 10 изображена часть функциональной схемы объекта контроля. При таком соединении выводов, элементы *D13* и *D14* представляют собой четырехразрядные счетчики–делители частоты на 16, а элементы *D16_2* и *D17_1* выполняют булеву функцию 2И–НЕ.

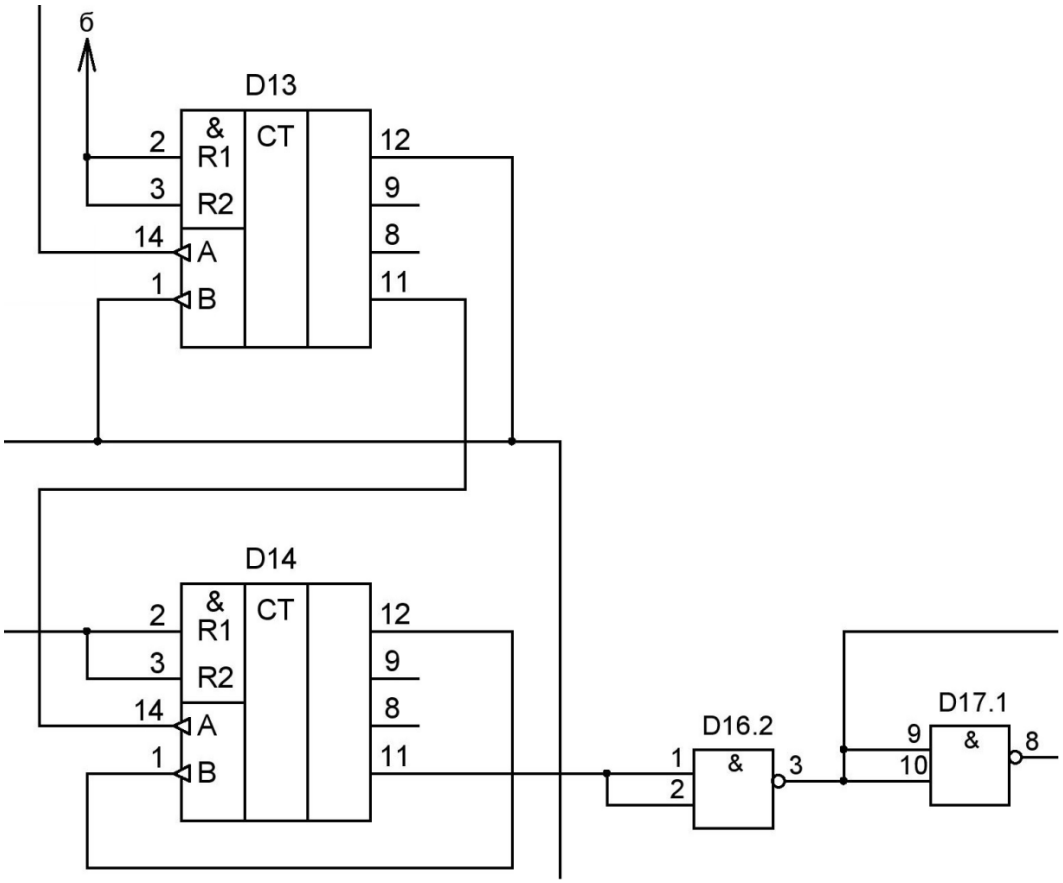

Рисунок 10. Фрагмент функциональной схемы объекта контроля.

На выходе с номером 11 элемента *D14* получается деление частоты поступающего на информационный вход *A* элемента *D13* сигнала на 256. Но это часть функциональной схемы. В устройстве подобным образом 5 счетчиков соединены в цепочку, уменьшающую частоту выходного сигнала в 1 048 576 раз по сравнению с частотой входного.

На рисунке 11 представлена часть файла результатов, сформированного системой «SimTest». Это временная последовательность работы последнего в цепочке из 5 счетчиков элемента *D14* и элемента И–НЕ *D19_3*, на входы которого подаются сигнал с выхода *q3* элемента *D14* и этот же сигнал, но прошедший через задерживающий элемент с рисунка 6.

На интервале 510 – 520 мс (правая часть рисунка) наблюдается переход выходов счетчика *D14* из состояния 1110 в состояние 0001 и подача сигнала с выхода *q3* на входы элемента *D19_3*. После этого, в момент 520 мс «приходит» импульс сброса, выходы счетчика устанавливаются в состояние 0000 и из-за задержки распространения сигнала через интегрирующий элемент на вход *in1* элемента *D19_3*, на выходе *out* этого элемента появляется отрицательный импульс, который доходит до выходов схемы *OUT_8* и *OUT_DIOD*, не изменяющие до этого своего состояния.

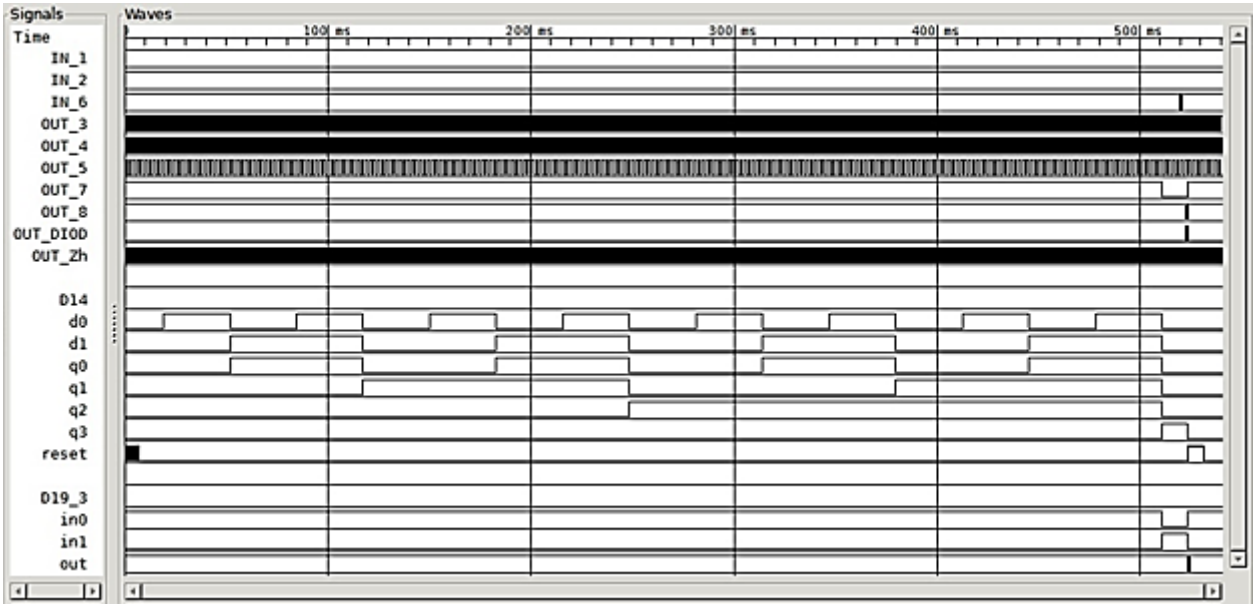

Рисунок 11. Временная диаграмма работы компонентов программной модели.

Так как тестовая программа содержит программные модели 37 логических элементов (элементы И–НЕ, триггеры и счетчики) и 8 аналоговых компонент (элементы с задержкой, повторители и инверторы сигнала, генераторы переменного сигнала), а также 72 сигнальные линии, то общее количество проверяемых объектов равно 109.

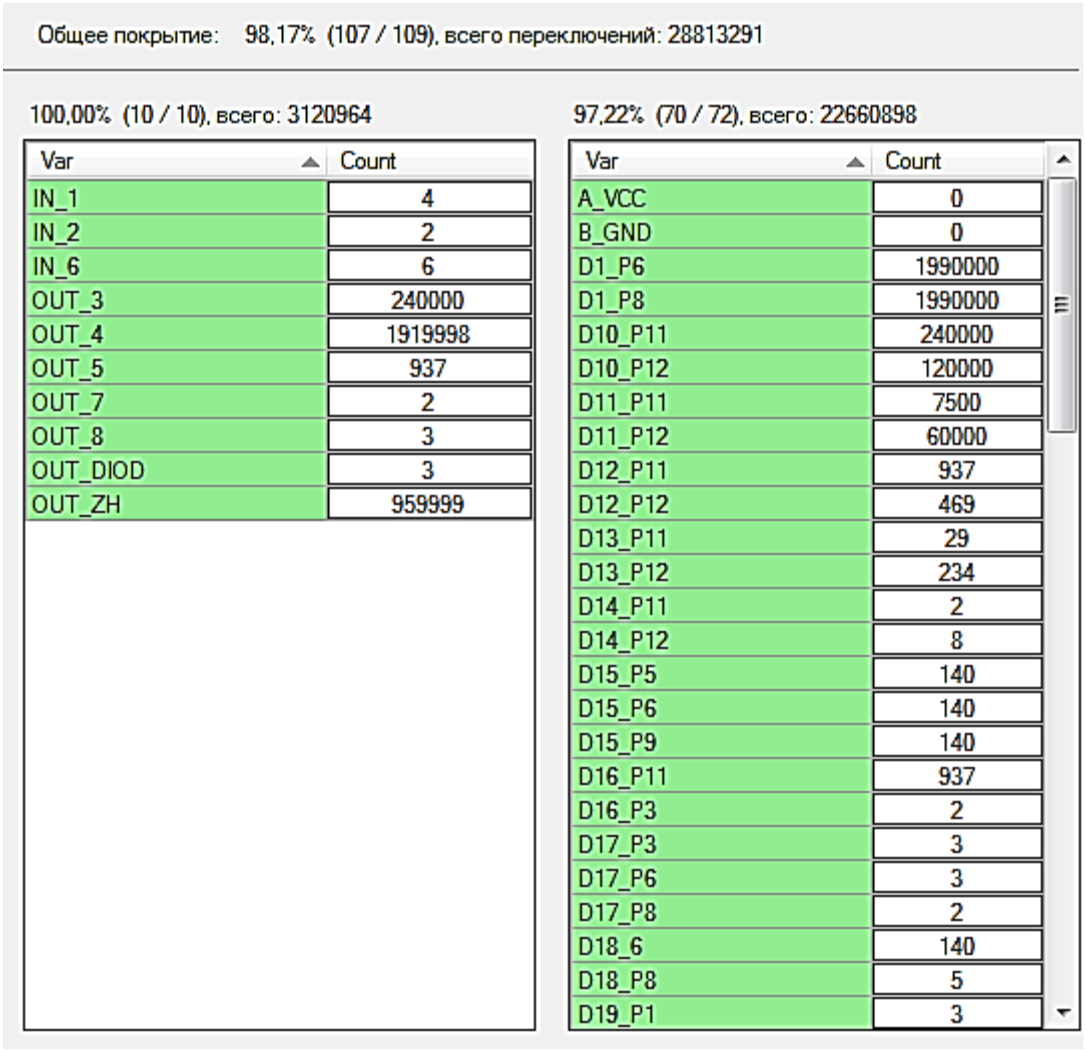

Рисунок 12. Анализ покрытия.

На рисунке 12 приведен анализ покрытия последовательностью тестовых воздействий. В левой части рисунка представлено количество переключений внешних выводов, а в правой — количество переключений внутренних сигнальных линий. Вверху отображена информация об общем покрытии 109 проверяемых объектов (45 элементов и 72 линий сигналов).

Так как сигнальные линии *A_VCC* и *B_GND* подключены к источникам постоянной логической единицы и постоянного логического нуля соответственно, то изменение их состояний невозможно. При удалении этих линий из статистики переключений общее покрытие составит 100% и, таким образом, будет являться полным.

Выводы

1. Разработаны и протестированы программные модели логических элементов 133LA2, 133LA3, 133LA4, 133LA7, 133LA8 (выполняют булевы функции 8И–НЕ, 2И–НЕ, 3И–НЕ, 4И–НЕ и 2И–НЕ соответственно), 133ТМ2 (содержит 2 идентичных синхронных D–триггера с дополняющими выходами), и 133IE5 (четырехразрядный двоичный счетчик), а также программные модели некоторых аналоговых элементов.

2. Создана программная модель радиоэлектронного устройства Субблок 1Э3 в среде автоматизированного проектирования AlteraQuartus 2.

3. Составленная при помощи системы автоматизированного проектирования тестов «SimTest» временная последовательность тестовых воздействий обладает полным покрытием объекта контроля и позволяет:

- проверить состояние работоспособности тестируемого устройства;
- диагностировать неисправности внутренних элементов;
- проверить целостность связей между элементами;
- определить отсутствие дополнительных «паразитных» связей внутри объекта контроля.

Таким образом, решены все поставленные задачи, и цель работы достигнута.

Заключение

В работе рассмотрено современное состояние научных исследований в области контроля и диагностики неисправностей радиоэлектронных устройств, достаточно подробно описаны этапы методики разработки тестовой программы при использовании технологии функционального тестирования.

Составленная последовательность тестовых воздействий обладает полным покрытием компонентов и сигнальных линий объекта контроля и удовлетворяет предъявляемым к тестовой программе критериям качества, что подтверждается представленными результатами функционального моделирования радиоэлектронного устройства.

Литература

1. Городецкий А. Снова о внутрисхемном тестировании ICT // Компоненты и технологии. 2011. №7. С. 58–59.

2. Albee A. J. The evolution of ICT: PCB technologies, test philosophies, and manufacturing business models are driving in-Circuit test evolution and innovations // IPC APEX EXPO Conference and Exhibition 2013, 1. P. 381–401.

3. Holtzer M. In-circuit pin testing: An excellent potential source of value creation // SMT Surface Mount Technology Magazine, 2015, 30 (6). P. 68–71.

4. Nelson R. Systems and software support PCB test // EE: Evaluation Engineering, 2013, 52 (2). P. 14–17.

5. Renbi A., Delsing J. Application of Contactless Testing to PCBs with BGAs and Open Sockets // Journal of Electronic Testing: Theory and Applications, 2015, 31 (4). P. 339–347.

6. Renbi A., Delsing J. Contactless Testing of Circuit Interconnects // Journal of Electronic Testing: Theory and Applications, 2015, 31 (3). P. 229–253.

7. Wang, R., Chakrabarty, K., Bhawmik, S. Interconnect testing and test-path scheduling for interposer-based 2.5-D ICs // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34 (1), art. no. 6936331. P. 136–149.

8. Ren X., Tavares V.G., Blanton R. D. S. Detection of illegitimate access to JTAG via statistical learning in chip // Proceedings – Design, Automation and Test in Europe, 2015, art. no. 7092367. P. 109–114.

9. Nelson R. JTAG and embedded test complement ATE // EE: Evaluation Engineering, 2014, 53 (3). P. 14–17.

10. Shashidhara H. B., Yellampalii S., Goudanavar V. Board level JTAG/boundary scan test solution // Proceedings of International Conference on Circuits, Communication, Control and Computing, 2014, art. no. 7057760. P. 73–76.

11. Yin X.H., Xu C.F. On a method of getting test data for boundary scan interconnection test in multiple scan chains // Advanced Materials Research, 2014, 986–987. P. 1531–1535.

12. Peng K. B., Zhang J. T. Reconfigurable boundary scan tester using cellular-automata register technology // Advanced Materials Research, 2014, 1006–1007. P. 986–989.

13. Wang R., Chakrabarty K., Eklow B. Scan-based testing of post-bond silicon interposer interconnects in 2.5-D ICs // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2014, 33 (9), art. no. 6879596. P. 1410–1423.

14. Deng X., Xu S., Zhang Y. An approach to generating test data sequences of boundary scan test system // Proceedings of 2013 IEEE 11th International Conference on Electronic Measurement and Instruments,2013, 1, art. no. 6743004. P. 264–270.

15. Sangi R., Baranski M., Oltmanns J., Streblow R., Müller D. Modeling and simulation of the heating circuit of a multi-functional building // Energy and Buildings, 2016, 110. P. 13–22.

16. Fujita M., Taguchi N., Iwata K., Mishchenko A. Incremental ATPG methods for multiple faults under multiple fault models // Proceedings – International Symposium on Quality Electronic Design, 2015, art. no. 7085420. P. 177–180.

17. Hobeika C., Thibeault C., Boland J.-F. Functional constraint extraction from register transfer level for ATPG // IEEE Transactions on Very Large Scale Integration 2015, vol. 23, 2, art. no. 6778092. P. 407–412

18. Kochte M. A., Elm M. b , Wunderlich H.-J. Accurate X-propagation for test applications by SAT-based reasoning // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, vol. 31, 12, art. no. 6349431. P. 1908–1919.

19. Bhowmik B., Deka J. K., Biswas S. Beyond test pattern generation: Coverage analysis // International Conference on Industrial Instrumentation and Control, 2015, art. no. 7151009. P.1620–1625.

20. Thoulath Begam V. M., Baulkani S. Compact test set method for high fault coverage test pattern generation // International Journal of Applied Engineering Research, 2015, vol. 10, 55. P. 453–458.

21. Matinnejad R., Nejati S., Briand L., Bruckmann T., Poull C. Search-based automated testing of continuous controllers: Framework, tool support, and case studies // Information and Software Technology, 2015, vol. 57, 1. P. 705–722.

22. Ghiduk A. S. Automatic generation of basis test paths using variable length genetic algorithm // Information Processing Letters, 2014, vol. 114, 6. P. 304–316.

23. Melnik V. I., Mikhailov A. N., Grishkin V. M., Ovsyannikov D. A., Yelaev Y. V. Methods of modeling of the test inputs for analysis the digital devices // International conference on computer technologies in physical and engineering applications, 2014. P. 112–113.

24. Grishkin V., Yelaev Y., Lopatkin G., Mikhailov A., Ovsyannikov D. Interface method of digital devices testing // Tenth International Vacuum Electron Sources Conference & Second International Conference on Emission Electronics, 2014. P. 107–108.

25. Гришкин В. М., Лопаткин Г. С, Михайлов А. Н., Овсянников Д. А. Интерфейсный метод построения моделей входных воздействий для тестирования электронных цифровых модулей // Вопросы радиоэлектроники, серия ОТ. 2013. № 1. С. 80–88.

26. Гришкин В. М., Степанов Ю. Л., Лопаткин Г. С., Большаков А. А. Подход к разработке тестов цифровых электронных модулей для автоматического тестового оборудования // Вопросы радиоэлектроники. 2013. Т. 1. № 1. С. 89–99.

27. Melnik V. I., Mikhailov A. N., Grishkin V. M., Ovsyannikov D. A., Yelaev Y. V. Modeling methods of the test inputs for analysis the digital devices //

2nd International Conference on Emission Electronics Selected papers. 2014. P. 48–50.

28. Михайлов А. Н., Мельник В. И., Овсянников Д. А. Тестовый контроль и диагностика радиоэлектронной аппаратуры // Электроника: Наука, технология, бизнес. 2013. № S (128). С. 114–117.

29. Елаев Е. В., Степанов Ю. Л., Ферсенков В. В. Подходы к моделированию микропроцессоров для построения контрольно-диагностических тестов // Процессы управления и устойчивость, 2015. Т. 2. № 1. С. 398–403.

30. Лопаткин Г. С. Подход к автоматизации тестирования электронных цифровых устройств // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. 2013. № 4. С. 90–98.

31. Мащинский Н. С., Елаев Е. В., Федюкович П. А. Моделирование сложных цифровых устройств с целью их тестирования // Процессы управления и устойчивость. 2015. Т. 2. № 1. С. 452–457.

32. Мельник В. И., Гришкин В. М., Михайлов А. Н., Овсянников Д. А. Методика разработки тест-программ контроля и диагностики цифровых устройств с использованием САПР «SimTest» // Электроника: Наука, технология, бизнес. 2013. № S (128). С. 118–124.

33. Степанов Ю. Л., Гришкин В. М., Елаев Е. В., Федюкович П. А. Развитие программной среды «Ястек» и ее использование при написании тестовых программ для цифровых модулей // Вопросы радиоэлектроники, 2015. № 2 (2). С. 198–205.

34. Степанов Ю. Л., Гришкин В. М., Большаков А. А., Лопаткин Г. С., Ким М. А. Автоматизированное построение тестов цифровых электронных модулей для комплекса тестового контроля и диагностики УТК-512 // Вопросы радиоэлектроники. 2012. Т. 1. № 1. С. 79–89.

35. Шило В. Л. Популярные цифровые микросхемы. М.: Радио и связь, 1987. 352 с.